Warmup

Open up Ripes and write a program that computes

```
(3 + 28 + 19) mod 4
```

Useful instructions:

```
li x0, 1 # load the value 1 into x0
add x3,x2,x1 # x3 = x2 + x1
andi x3,x2,0xF0 # x3 = x2 & 0x000000F0 (32 bit AND)
```

Mod is hard in general, but easy for powers of 2!

EE 201: Memory and function calls in RISC-V

Steven Bell

28 March 2024

Tufts UNIVERSITY

By the end of class today, you should be able to:

Be comfortable writing assembly code that does "real stuff".

- Use `lw`/`sw` instructions to read and write memory
- Use the `jal` and `jalr` instructions to implement function calls

Load memory (into register)

Load the (32-bit) word at rs1+offset into rd.

```
lw rd, offset, rs1
```

Store register into memory

Store the (32-bit) word in rs2 to the memory at rs1+offset

```
sw rs2, offset, rs1
```

Practice time!

Write an assembly program which reads a value from memory address 0x100, adds 10 to it, and stores the result back.

Functions

To call a function:

Jump to the the first instruction in the function

When the function returns:

Functions

To call a function:

Jump to the the first instruction in the function

And remember where to continue from (ra, aka x1)

When the function returns:

Functions

To call a function:

- Put the arguments into the registers (a0-a7) + stack
- Jump to the the first instruction in the function
- And remember where to continue from (ra, aka x1)

When the function returns:

- Retreive the result from a0

Functions

To call a function:

- Save any registers that might be overwritten
- Put the arguments into the registers (a0-a7) + stack
- Jump to the the first instruction in the function
- And remember where to continue from (ra, aka x1)

When the function returns:

- Retreive the result from a0
- Restore registers as necessary

Practice time

Write a function that computes the absolute value of a number, and call it from your main code.